Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»
Курс «Базовые компоненты интернет-технологий»

Отчёт по лабораторной работе №5

«Модульное тестирование в Python»

Выполнил:
студент группы ИУ5-33Б
Рыбин Владислав

Проверил:
к.т.н., доц., Ю. Е. Гапанюк

2022 г.

Задание:

1. Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
2. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
3. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
    - TDD - фреймворк (не менее 3 тестов).
    - BDD - фреймворк (не менее 3 тестов).

Текст программы:

Field_test.py

```
import unittest
from lab_python_fp.field import field

class TestField(unittest.TestCase):
    def setUp(self):
        self.goods = [
           {'title': 'Ковер', 'price': 2000, 'color': 'green'},
           {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
        ]

    def test_one_argument(self):
        result = list(field(self.goods, 'title'))
        answer = ['Ковер', 'Диван для отдыха']
        self.assertEqual(result, answer)

    def test_many_arguments(self):
        result = list(field(self.goods, 'title', 'price'))
        answer = [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для
отдыха', 'price': 5300}]
        self.assertEqual(result, answer)


if __name__ == '__main__':
    unittest.main()
```

test_unique.py

```
import unittest
from lab_python_fp.unique import Unique
from lab_python_fp.gen_random import gen_random

class TestUnique(unittest.TestCase):
    def test_numbers(self):
        data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
        result = list(Unique(data))
        answer = [1, 2]
        self.assertEqual(result, answer)

    def test_random_generator(self):
        data = gen_random(10, 1, 3)
        result = set(Unique(data))
```

```
        answer = set(range(1, 4))
        self.assertTrue(answer.issubset(result))

    def test_letters(self):
        data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        result = list(Unique(data))
        answer = ['a', 'A', 'b', 'B']
        self.assertEqual(result, answer)

    def test_letters_ignoring_case(self):
        data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        result = list(Unique(data, ignore_case=True))
        answer = ['a', 'b']
        self.assertEqual(result, answer)


if __name__ == '__main__':
    unittest.main()
```

Пример выполнения